Simulation Performance Optimization
Using Multivariate Minimization Techniques

James C. Eamon & Dennis R. Ellis
Joint National Test Facility CRDC/TISA
730 Irwin Avenue, Schriever AFB, CO 80912
jeamon@jntf.osd.mil
dellis@jntf.osd.mil

Keywords:
Performance, wargame, simulation, SPEEDES, minimization, PDES

ABSTRACT

The current generation of wargame simulations under development at the Joint National Test Facility must provide real-time performance in large-scale Theater Air Missile Defense exercises. These simulations are based on discrete event simulation frameworks, run in a parallel mode across multiple processors, and employ optimistic time management methods to improve performance and synchronize event processing. The performance of these simulations is dependent not only on the size of the scenario being studied, but also on the manner in which the simulation processes are distributed among the multiple processors and the various tuning parameters that control the time management techniques. This paper describes a study that explored the use of numerical multivariate minimization techniques to optimize these tuning parameters in order to enhance the simulation performance. Similar techniques were applied to determine if an optimum method could be found for distributing the simulation processes across multiple processors. An additional benefit of the study was the evaluation of the feasibility of dynamically adjusting the time synchronization parameters during the course of the simulation.

INTRODUCTION

Wargame 2000 (WG2K) is a real-time simulation under development at the Joint National Test Facility (JNTF) to evaluate future ballistic and theater missile defense architectures, defense doctrine, and concepts of operation. The performance requirements for WG2K are much more demanding than those of previous simulations in terms of the number of entities that are to be modeled, the level of fidelity, and the speed at which the models must run. It is estimated that in some scenarios, WG2K will have to process hundreds of thousands of events per second under peak loads while meeting real-time constraints. The WG2K developers are exploiting several innovative simulation techniques to meet the demanding performance requirements, including:

- Processing simulation events in a parallel fashion by distributing them simultaneously across multiple processors,
- Using "discrete event" (rather than discrete time) simulation methods, and
- Employing optimistic synchronization strategies for event flow control

These techniques offer the opportunity for substantial gains in efficiency and scalability, but as one might expect, there is some risk that their misuse may limit the gains that might otherwise be achieved. Unfortunately, the guidelines for implementing the methods in wargame simulation applications are not well defined. As experience is gained in their use, it is expected that these guidelines will become more apparent.

The Technology Insertion Studies and Analysis (TISA) group at the JNTF is investigating how performance is affected by scenario size, distribution of processes across processors, and the values used to "tune" the time management parameters that control event synchronization. We have explored several methods to enhance WG2K performance and provide automated techniques for determining the optimum processor allocation and time management tuning parameter values. The study used the same Parallel Discrete Event Simulation (PDES) framework used by WG2K, namely the Synchronous Paral-

F05604-95-D-9001

lel Environment for Emulation and Discrete Event Simulation (SPEEDES) [1].

BACKGROUND

SPEEDES implements the distribution of processes across processors and the event synchronization among nodes. It provides event flow control in terms of memory usage, anti-messages, and the aggressiveness of the optimistic event processing. The algorithm used to implement the optimistic event processing and synchronization in SPEEDES is called "Breathing Time Warp" (BTW) [2]. It is an extension and merger of two other optimistic time management algorithms: Time Warp [3] and Breathing Time Buckets (BTB) [4]. It is claimed that Breathing Time Warp has the efficiency gains of the Time Warp scheme, while avoiding the risks that the simulation gets bogged down in rollbacks and anti-messages. The phases of a typical processing time cycle in the Breathing Time Warp time management scheme implemented in SPEEDES are shown in Figure 1.

The BTW scheme processes events in time cycles like the BTB except that the cycle times are not constant, but adapt to the processing load. Every cycle starts in the risky Time Warp mode where the first N_{gvt} events beyond Global Virtual Time (GVT) have their messages sent out immediately. GVT is the minimum time tag of any unprocessed event in the queue. This means that these messages might have to be cancelled if this node subsequently receives a message with a time stamp earlier than one of those sent out. Since the messages sent out are close in time to GVT, the likelihood of them having to be recalled is small. When (and if) the number of events reaches N_{gvt}, SPEEDES switches to the less risky Breathing Time Buckets phase.

In the BTB mode, a "local event horizon" for each node is determined as the time stamp of the earliest new event generated in the current cycle on that node. Events processed beyond this time may have to be rolled back. The global event horizon is the minimum of all the local event horizons over all nodes. Events are processed optimistically but the messages they generate are not sent out until the global horizon is reached, so that the processing and messages are sent out risk free (the processes will never have to be rolled back and the messages will never have to be cancelled with anti-messages). When the global event horizon is reached, all nodes stop their processing and a GVT update is done.

Following the GVT update phase, messages with time tags less than GVT that have not yet been sent are released (committed) and messages are flushed out of the communications hardware.

Four input parameters are used in SPEEDES to implement the event synchronization and control [5]. These parameters (implemented in the Breathing Time Warp algorithm) include:

T_{gvt}: Maximum seconds between GVT updates (in terms of wall clock time).

N_{gvt}: The number of uncommitted events processed before requesting a GVT update.

N_{risk}: The number of events processed beyond GVT by each node that are allowed to send their messages with risk.

N_{opt}: The number of events allowed to be processed on each node beyond GVT.

The values chosen for these parameters (and relations between them) effect the number of rollbacks and anti-messages, thus directly impacting performance.

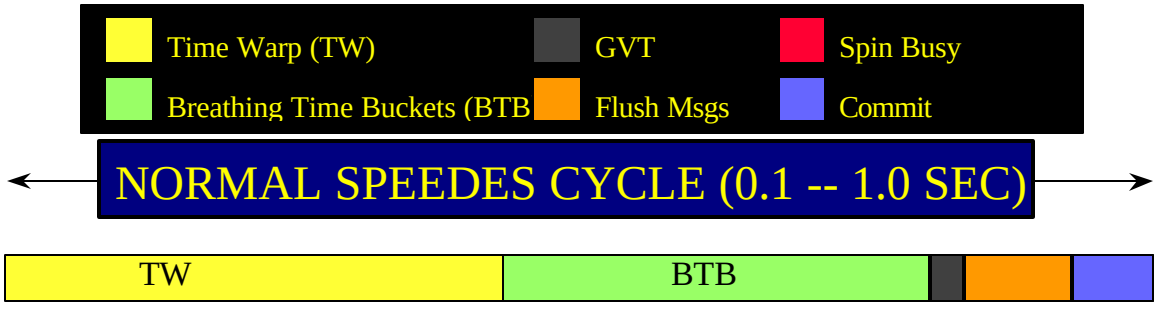

Figure 1 – Breathing Time Warp event synchronization phases in SPEEDES

There is a widely held belief (based on presentations and discussion with the SPEEDES developers) that over a wide range of values, changes in these parameters have little effect on SPEEDES performance (the so-called "bathtub" effect). In light of the general nature of these guidelines, most SPEEDES users are inclined to use the default values for all problems.

This paper describes a study undertaken to explore the use of numerical multivariate minimization techniques to optimize these tuning parameters in order to enhance the simulation performance. A follow-on study is underway to use similar technique to find the optimum number of processors across which the simulation should be distributed. Finally, the feasibility of dynamically adjusting the time synchronization parameters during the course of the simulation will be evaluated.

Since the WG2K system was not expected to exist by the time the study was to start, we needed to develop tools to evaluate WG2K performance under various synthetic loads. These included the Aggregated Wait Time Model (AWTM) [6] and a synthetic load builder tool called ThreadBuilder (TB) [7]. Using the AWTM, an analyst can quickly and easily generate a particular scenario of a future simulation by specifying only a few top-level parameters (number of threat missiles, aircraft, command centers, interceptors, sensors, etc.) of a wargame. The AWTM output is then reformatted into a form that can be processed by SPEEDES and input into the ThreadBuilder synthetic load generator which can estimate the performance of WG2K for that scenario.

This tool suite was used to analyze WG2K performance in a number of threat-sensor loading scenarios (Figure 2) that are representative of typical NMD scenarios. The scenario described in a simple, unclassified, and generally realistic manner the deployment time line of a threat missile attack, sensor detection, interceptor deployment, and engagement of the threat objects. This scenario and some variants of it were processed using the ThreadBuilder synthetic load generator coupled with SPEEDES. This process allowed a carefully controlled method of estimating how WG2K may perform under similar loading conditions.

In this performance evaluation, it was first necessary to choose an appropriate measure of merit from a number of possibilities. An obvious choice is the elapsed wall clock time for the run. Other choices include the number of rollbacks, the CPU time utilized, and what is referred to as the Simulation Time Advancement Rate or STAR which is a measure of SPEEDES performance that is directly tied to WG2K requirements. It indicates of how fast simulation time is advancing with respect to wall clock time. At a particular time in the simulation, for example, if the STAR value is two, the simulation is running twice as fast as real time. A STAR value of 0.5 indicates that the simulation is running at one-half real time (i.e, slower than real time). In the

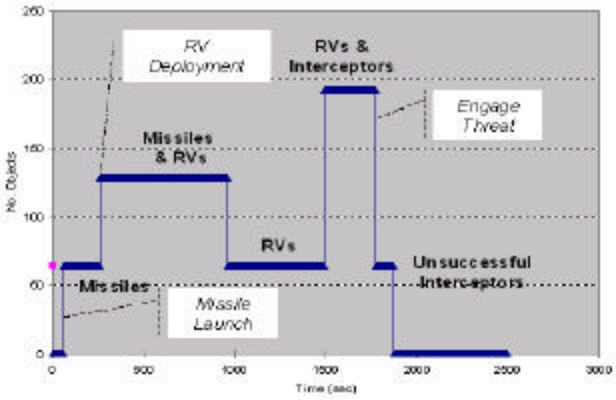

Figure 2 – Scenario Timeline

NMD scenario analysis runs for WG2K, the STAR plots were one of the more useful overall indicators of TB/SPEEDES performance for a number of stressing scenarios.

The results of our runs show that the sensor component of the NMD Aggregated Wait Time Model is the most demanding of processor time. The CPU loading depends directly on the number of threat objects seen by the sensor. The STAR performance indicator generally follows the simulated NMD sensor loads shown in Figure 2 (STAR varies inversely with loading). When the sensors were lightly loaded (e.g. reporting few threat objects), STAR values were large, often greater than 10, indicating the simulation was running at 10 or more times real time. As the scenario progressed through different stages and the sensors began reporting more objects (with the concomitant additional processing time and message traffic), the STAR values generally followed the number of threat objects viewed by the sensors.

For the lighter load cases (which were run on a single processor), the STAR indicator generally

followed the loading as expected. However, for heavier loads (four times as many threats) run on multiple processors, the STAR often showed erratic behavior. In addition, in some of these cases the number of rollbacks did not track with the STAR in a predictable fashion, as one would expect. Moreover, several attempts to characterize the sensitivity of TB/SPEEDES performance as a function of the time management tuning

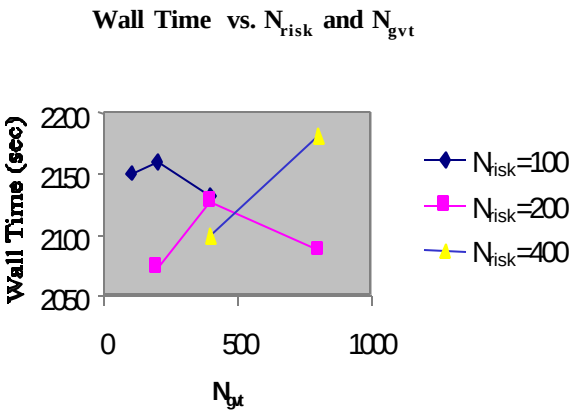

Figure 3 – Wall clock time vs. tuning parameters Ngvt and Nrisk for a NMD sce-

parameters were inconclusive. Figure 3, for example, is typical of the type of data generated in this series of runs. Shown in the figure are several curves showing how wall clock time varies with the tuning parameter Ngvt at three values of parameter Nrisk. It is not apparent what information about the proper choice of these parameters can be gleaned from these runs.

Since there was little clear and useful information that could be derived from this series of runs, a more controlled study was planned to systematically evaluate TB/SPEEDES performance sensitivity to the tuning parameters.

APPROACH

The approach we used for evaluating SPEEDES performance is illustrated in Figure 4. It used standard numerical minimization methods such as those described in Reference 8. We initially used the PROX missile defense model as the SPEEDES application instead of ThreadBuilder for the investigations of performance sensitivity with tuning parameters. For the analysis of performance sensitivity with scenario loading, we will use ThreadBuilder and SPEEDES. Rather than simply running the SPEEDES application over a wide range of tuning parameter values and trying to interpolate what the optimum values of the parameters should be, we instead used numerical methods to generate the combination of tuning parameters for a series of runs that progressively optimized SPEEDES performance. As each run was completed, the resultant value of some measure of merit (such as minimum STAR or Wall Clock Time) was compared with the

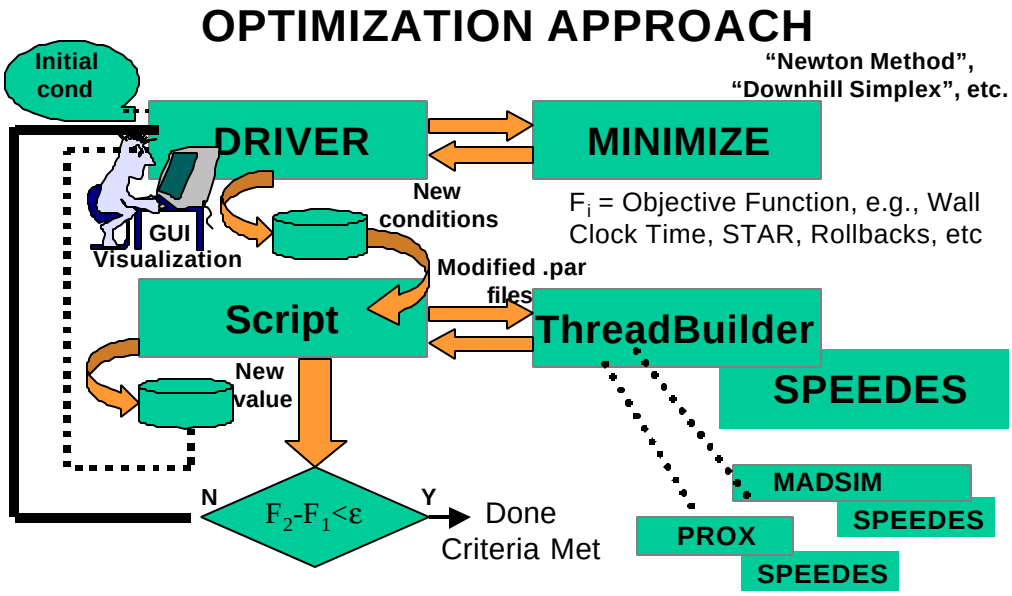

Figure 4 – An automated approach for determining the optimum tuning values

results using an earlier combination of parameters. If the newer parameters yielded significantly better (or worse) results, the minimization process was called to recommend another set of tuning parameters to try. When there is little or no difference in the results between runs for different parameters, the process terminated with the "optimum" values for the tuning parameters.

PRELIMINARY RESULTS

In order to test the feasibility of this approach, a prototype version was implemented using algorithms extracted from Reference 8. The method chosen was the "Multivariate Downhill Simplex" algorithm. A feature of this particular algorithm is that it uses only the value of the objective function (e.g., the wall clock time for the run) at different points and does not require the first

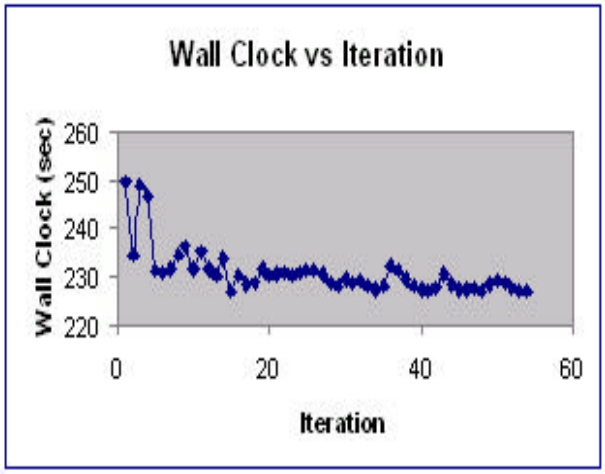

Figure 5 – Objective function (wall clock time) at each iteration of the optimization

derivative of the function. A simple driver program was written in the manner shown in Figure 4 to call the minimization routine and change the value in the ".par" file that PROX/SPEEDES uses. The routine would then recommend new values to try and the process was repeated until no difference in performance was achieved between runs. The results of one of these optimization runs are shown in Figures 5 – 6 for a scenario that was comprised of 8 missiles and 30 defense sensors. The optimization run was initiated with the following values of SPEEDES tuning parameters:

- $N_{gvt} = 25$
- $N_{risk} = 100$
- $N_{opt} = 500$

Using this initial set of parameters, the wall clock time used by the PROX/SPEEDES run was 250 seconds (Figure 5).

Figure 6 shows how the intermediate values of the three tuning parameters varied with each iteration as the numerical algorithm sought to maximize performance by minimizing the Wall Clock time. At the conclusion of the optimization process, after 58 iterations, the Downhill Simplex method had arrived at a set of "optimum" tuning parameters consisting of:

- $N_{gvt} = 406$
- $N_{risk} = 358$

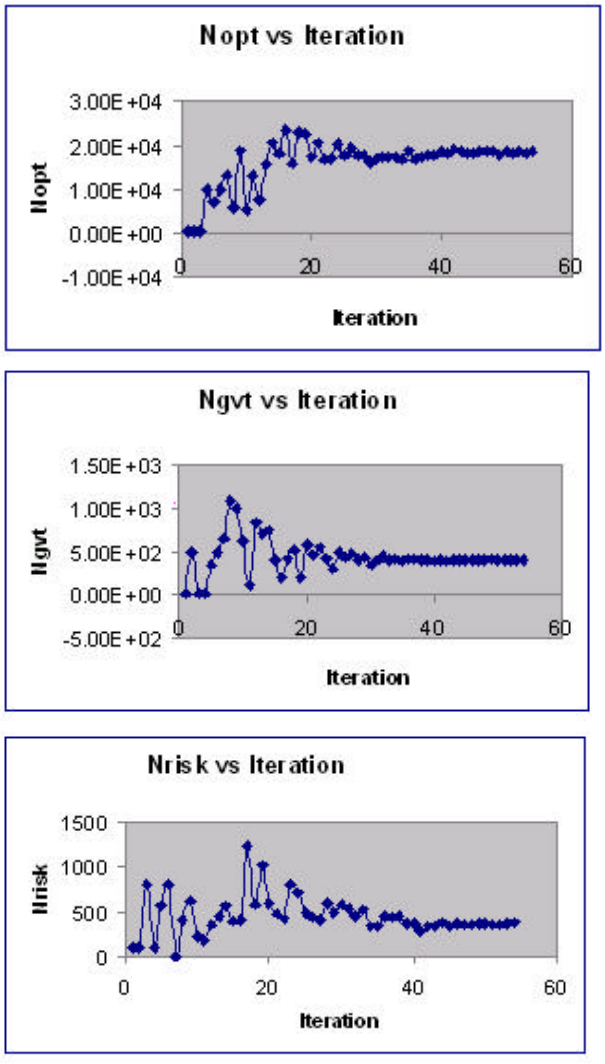

Figure 6 – The values of the tuning parameters recommended by the Downhill Simplex minimization routine

- N_{opt} = 18,357

The final PROX/SPEEDES run using these parameters took 226 seconds. By comparison with the initial run, a 10% reduction in wall clock time had been achieved for this relatively light loading scenario. It seems likely that greater performance increases can be gained for higher (and more realistic) TAMD scenario loads.

DISCUSSION

This initial attempt to determine if choosing better values of the time management tuning parameters can optimize SPEEDES performance produced promising results. We intend to expand the effort to look at other scenario loads and other minimization algorithms. We plan to implement a modified version of the Newton methods, a simulated annealing algorithm, and others.

We also plan to expand the analysis to include the affect of distributing the load across different numbers of processors. Intuitively, we expect there is an optimum number of processor to use for each scenario loading. We further expect that if that number is used, large performance benefits can be achieved (greater than the 10% we have seen so far for optimizing the tuning parameters).

PRODUCTS

This task has, and will continue to, produce a number of important products that will benefit WG2K developers. These include:

- A better understanding of MADSim/SPEEDES performance sensitivity with the time synchronization parameters
- An understanding of how scenario loading relates to time management parameters and number of processors in determining SPEEDES performance
- An evaluation of the utility of several numerical minimization techniques in this application
- A determination of the feasibility of dynamically optimizing the time management parameters (i.e., during a single MADSim/SPEEDES run)
- A tool for enhancing MADSim/SPEEDES performance prior to a wargame for a defined scenario by optimizing time management parameters and choosing the number of processors to use
- A report describing the results of the task
- A demonstration capability that will show the benefits of the optimization method

ACKNOWLEDGMENTS

Pat Talbot's ideas on performance optimization were the genesis of this task. Dennis Ellis wrote the necessary scripts and driver program to implement the approach. The assistance of Frank Deis in understanding the concepts of parallel discrete event simulations and performance metrics is gratefully acknowledged.

REFERENCES

1. Steinman, Jeff S., "SPEEDES: A Multiple-Synchronization Environment for Parallel Discrete-Event Simulations." International Journal in Computer Simulation, Vol. 2 Pages 251-286
2. Steinman, Jeff S., "Breathing Time Warp", Jet Propulsion Laboratory, unpublished paper contained in SPEEDES documentation package.
3. Jefferson, D., "Virtual Time." *ACM Transactions on Programming Languages and Systems*. Vol. 7, No. 3, Pages 404-425, 1985.
4. Steinman, Jeff S., "Multi-Node Test Bed: A Distributed Emulation of Space Communications for the Strategic Defense System." Proceedings of the Twenty-First Annual Pittsburgh Conference on Modeling and Simulation, Vol. 21, Part 3. Pages 1111-1115, 1990.
5. Steinman, J., Lee, C., Wilson, L., and Nicol, D., "Global Virtual Time and Distributed Synchronization", unpublished paper contained in SPEEDES Documentation package
6. Eamon, J., *"Theater Missile Defense Wargame Performance Requirements and Analysis Models*," TISA 1999 Research Plan, March 1999
7. Eamon, J., *"Theater Missile Defense Wargame Performance Requirements and Analysis Models*," TISA 1999 Research Plan, March 1999
8. Numerical Recipes in FORTRAN: the art of scientific computing, William H. Press et. al., 2nd edition, Cambridge Press, 1992